(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 115 061 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.07.2001 Bulletin 2001/28

(51) Int Cl.[7]: **G06F 9/50**, H04L 29/06

(21) Application number: **00310628.3**

(22) Date of filing: **30.11.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **30.11.1999 US 452286**

(71) Applicant: **Ensim Corporation**
Sunnyvale, California 94089 (US)

(72) Inventors:
- **Goyal, Pawan**
  Mountain View, California 94043 (US)
- **Huang, Xun Wilson**
  Mountain View, California 94043 (US)
- **Keshav, Srinivasan**
  Mountain View, California 94303 (US)
- **Sharma, Rosen**
  Mountain View, California 94043 (US)

(74) Representative:
McLeish, Nicholas Alistair Maxwell et al
Boult Wade Tennant
Verulam Gardens
70 Gray's Inn Road
London WC1X 8BT (GB)

(54) Providing quality of service guarantees to virtual hosts

(57) A method facilitates providing appropriate quality of service guarantees to a plurality of virtual hosts on a single physical host computer. A server application program and its child processes service communication requests made to the plurality of virtual hosts. Quality of service parameters associated with the virtual hosts are stored. Communication requests made to a specific one the virtual host are detected. The quality of service parameters associated with the specific virtual host are obtained. Operating system resources are utilized to guarantee, to a child process of the server application program, a quality of service according to the obtained quality of service parameters associated with the virtual host. Communication between the virtual host and the client is allowed to proceed, the communication being managed by the child process.

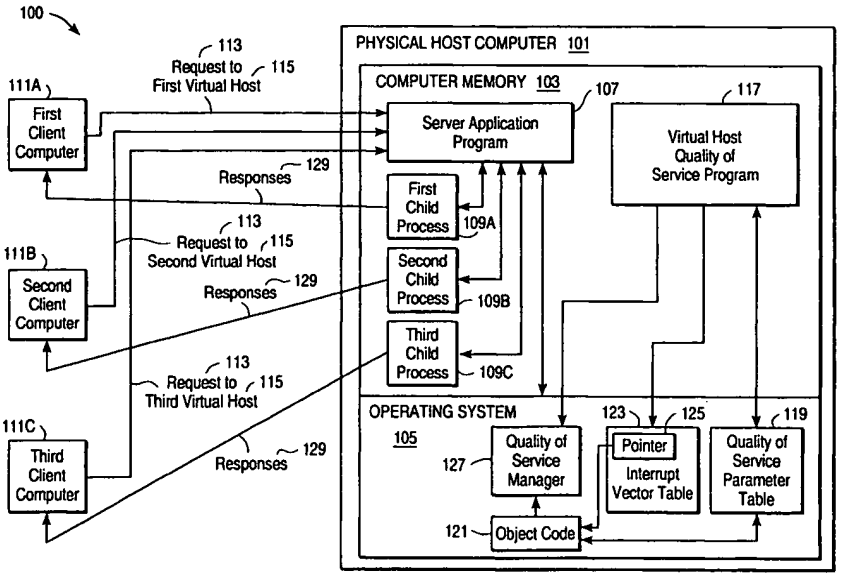

FIG. 1

EP 1 115 061 A2

Description

BACKGROUND

FIELD OF INVENTION

[0001] The present invention relates generally to guaranteeing appropriate quality of service to virtual processes executing on multitasking operating systems, and specifically in one embodiment to guaranteeing quality of service to virtual hosts servicing client requests to multiple network addresses on a single physical host computer.

BACKGROUND OF INVENTION

[0002] With the popularity and success of the Internet, server technologies are of great commercial importance today. Typically, a server program executing on a single physical host computer services client requests made to a single network address allocated to the host. However, using Transmission Control Protocol (TCP) and other transport protocols, a server application executing on a single physical host can be programmed to process requests made to multiple network addresses. Such functionality is known as virtual hosting.

[0003] To enable virtual hosting, more than one network address is assigned to a single physical host computer. A server program executing on the host opens a communication transport channel (socket) and allows receipt of incoming communications targeted for any of the multiple network addresses assigned to the host. Accepting a communication request by a server executing TCP is a three step process that comprises waiting for the communication request from a client, sending an acknowledgment signal to the client, and receiving a return acknowledgment signal from the client. This three step process is called "three way handshaking," and is a feature of TCP communication. While accepting a communication request, a server is unable to execute other tasks. Thus, a server can not wait for one incoming communication request and simultaneously service a separate communication request. However, multiple communication requests made to a plurality of network addresses associated with a single physical hosts require simultaneous service. For this reason, a virtual host server typically accepts the communications requests itself, and creates child processes to service the requests.

[0004] Two known methods exist for utilizing child processes to service communication requests. The most common involves accepting the communication request by the server, and then generating (forking) a child process to service it. While the child process services the request made by the client and transmits it to the client, the server is free to accept the next communication request, perhaps from another client. As soon as the server accepts a request, the server generates a new child process, which services the request and then terminates. This method of virtual hosting, know as "fork after accept," is widely used today.

[0005] Another method of virtual hosting comprises generating, by the server, a plurality of child processes. The number of child processes created reflects a desired maximum number of communication requests to service simultaneously. Each child process proceeds to accept a communication request. Thus, each child waits for an incoming communication request, services it, and proceeds to wait for a next request. While a child process is servicing a request it can not be waiting for another request, but this is allowable because other child processes are waiting for incoming requests. Where all the child processes are busy servicing requests, the maximum desired number of requests are being serviced, and no more can be accepted until one of the child processes finishes servicing a request, and begins waiting for a new one. This method, called "fork before accept," is know and used today, although less commonly than "fork after accept."

[0006] Both virtual host systems that utilize the fork before accept method and those that utilize the fork after accept method can be name-based as well as address-based. As described above, in an address-based virtual host system, each virtual host is identified by a separate network address assigned to a single, physical host. In contrast, in a name-based virtual host system, each virtual host is identified not by a network address, but instead by a domain name. Multiple domain names can be assigned to a single address. Thus, multiple virtual hosts, each identified by a unique domain name, can all be assigned to a single network address. The single network address to which the multiple domain names are assigned is assigned to the single, physical host. Communication requests to a name-based virtual host are made to a domain name, not to a network address. As with address-based virtual host systems, all virtual hosts map to a single physical host.

[0007] A virtual host system simulates multiple hosts by servicing client requests made to any of the multiple network addresses or domain names. This is desirable, because providing a unique physical host for each network address or domain name is expensive and inefficient. Hosting services are often provided commercially by an Internet Service Provider (ISP). Without virtual hosting, an ISP would have to provide a separate physical host computer with a unique network address for every customer that purchases host services. Often, a customer purchasing host services from an ISP will neither require nor be amenable to paying for use of an entire host computer. Generally, only a fraction of the processing power, storage, and other resources of a host computer will be required to meet the needs of an individual customer.

[0008] Virtual hosting allows an ISP to utilize one physical host computer to provide commercial host services to multiple customers, thereby creating a virtual

host server. Each customer is assigned a network address (or domain name), and is provided with resources on the single, physical host computer, effectively sharing the host with other customers. A client computer requests data from a specific customer's host by targeting communication requests to the appropriate network address (or domain name). By utilizing the fork before accept method or the fork after accept method, the virtual host server can service requests to multiple network addresses or domain names. Thus, the functionality of numerous hosts is provided by a single physical host computer, servicing requests made to a plurality of network addresses and domain names by multiple customers.

[0009] One shortcoming with virtual hosting as it exists today is the inability to allocate appropriate amounts of computer resources of the physical host to servicing client requests made to specific virtual hosts, and hence to specific customers. Where an ISP provides host services to multiple customers on a single physical computer, it is desirable to allot to each virtual host a specific amount of computer resources appropriate to the needs of the customer, and preferably based upon the amount paid for the services. For example, suppose two customers purchase host services from an ISP. The first customer is a large corporation providing financial services to thousands of clients internationally. The financial services host requires a great deal of storage space, as well as prompt response time to all client requests. Of course, the first customer is willing to compensate the ISP appropriately for providing such a level of host services. The second customer is a sole proprietorship that sells floral arrangements locally. The second customer has a very limited budget, but only requires minimal computer resources. Clearly, it is desirable for the ISP to allocate different percentages of the system resources to the two separate virtual hosts provided by the ISP for the two separate customers. However, this is not possible with conventional virtual hosting techniques.

[0010] Multitasking operating systems such as UNIX facilitate specification of resource allocation to multiple concurrent processes. The operating system can be instructed as to how to allocate resources to different processes. System resources can be allocated to processes as a percentage of resources available (for example, the operating system may be instructed to allocate twenty percent of the central processing unit cycles to process A and two percent to process B), or as specific numbers of units (for example, the operating system may be instructed to allocate X cycles per second to process A and Y cycles per second to process B). Such specification of resource allocation is called a guarantee of quality of service.

[0011] A server, which is a process, executing on a dedicated physical host services client requests for a single network address (physical host) only. Thus, quality of service can simply be set for the server to the quality of service appropriate for the host. A virtual host server services numerous client requests for multiple virtual hosts. A single virtual host server provides host services for a plurality of customers all of whom may require different quality of service. Although it would be possible to set a single quality of service for the virtual host server, no single quality of service is appropriate for all of the virtual hosts. Because the appropriate quality of service for different virtual hosts is different, providing a single quality of service for all virtual hosts is undesirable. As detailed above, ISP's utilize single server application programs to provide virtual hosting services to multiple customers with varying business needs and budgets. It would be desirable for an ISP to be able to make appropriate quality of service guarantees to different customers purchasing virtual host services. However, conventional virtual host systems can provide only the same quality of service for all virtual hosts supported by the server. With existing virtual host systems, an ISP cannot provide one quality of service to the international financial services corporation of the example given above, and another to the local florist. The ISP either must provide each customer with the same quality of service, which is undesirable, or provide each customer with a separate physical host, which is inefficient and expensive for the ISP, as many customers do not require the resources of a dedicated host computer. What is needed is a virtual host system that provides appropriate quality of service guarantees for each virtual host.

[0012] Virtual host systems are being utilized today by ISP's and other providers of host services. As explained above, it would be desirable for existing ISP's providing virtual host systems to be able to provide appropriate quality of service guarantees to multiple virtual hosts. However, many such ISP's would not want to replace their existing virtual host system with one that could provide such guarantees, even if such a system were available. Upgrading a virtual host system is a time consuming and complicated process, often involving costly down time and high labor expenses. It would be desirable to have a system to allow existing users to provide appropriate quality of service guarantees to multiple virtual hosts, without having to upgrade or replace their existing virtual host systems.

SUMMARY OF INVENTION

[0013] The present invention allows providers of virtual host services to make appropriate quality of service guarantees to multiple virtual hosts. In one embodiment, the present invention executes as a virtual host quality of service application program on the same physical host computer as an existing virtual host server. The application program modifies the operating system of the host computer to include a quality of service table comprising appropriate quality of service parameters for all virtual hosts. Then, object code is inserted into the operating system. The object code comprises computer instructions to obtain quality of service parameters for a particular virtual host from the table, and to set the qual-

ity of service for the virtual host according to these parameters. Once the object cod is inserted, system calls to the operating system pertaining to communications between one of the virtual hosts and a client are intercepted. The system calls are intercepted by replacing a pointer in an operating system function table with a pointer to the inserted object code, so that when a system call is made, the inserted object code is executed rather than the system call. The object code obtains quality of service parameters for the selected virtual host from the quality of service table, and the appropriate quality of service is guaranteed to the virtual host. Virtual host systems utilize child processes of a parent server application to manage communications between a virtual host and a client. Therefore, the inserted object code can guarantee the appropriate quality of service to the virtual host by setting the quality of service guarantees for the child process managing the communication between the virtual host and the client. After the quality of service guarantees are set, the communication between the virtual host and the client proceeds.

[0014] The present invention works for all types of virtual hosting systems, including both those that utilize the fork before accept method and those that utilize the fork after accept method. To set quality of service guarantees for a virtual host system utilizing the fork before accept method, system calls that establish a communication channel between a client and a select one of the virtual hosts are intercepted. These system calls are made by child processes of the server application. The present invention then guarantees quality of service appropriate for the virtual host to the child process that made the intercepted system call. Once the quality of service is guaranteed, the child process that made the system call manages the communication between the virtual host and the client.

[0015] Guaranteeing quality of service to virtual hosts of a system utilizing the fork after accept method is similarly facilitated. In a system utilizing the fork after accept method, a communication channel is established by the server application prior to generation of a child process to service a communication request. At the time a communication channel is established, it is unknown what child process will service that communication channel. Therefore, under these circumstances, the virtual host quality of service application program intercepts system calls, made by child processes, that transport data through already established communication channels. When such a system call is made, the virtual host quality of service program determines the virtual host to which the client is communicating via the channel. Appropriate quality of service is then guaranteed to the child process that made the intercepted system call. The child process then proceeds to manage communication between the virtual host and the client.

[0016] The present invention also guarantees appropriate quality of service to name-based virtual hosts. Clients make communication requests to name-based virtual hosts by passing a domain nam to the virtual host server. The domain name is then stor d in a communication buffer utilized for communications between th server and the client. The virtual host quality of service program int rcepts system calls that transport data through communication channels. When such a system call is made, the communication buffer being used to transport data through the channel is parsed. By parsing the buffer, the domain name to which the client is communicating is isolated, and hence the virtual host associated with the communication is identified. Quality of service guarantees appropriate for the virtual host are then set for the child process that made the system call. That child process proceeds to manage the communication between the virtual host and the client.

[0017] In all of the above described embodiments, the present invention allows providers of virtual host services to make appropriate quality of service guarantees to multiple virtual hosts without upgrading or replacing existing virtual host systems.

[0018] In another embodiment, the present invention comprises a stand alone virtual host system that sets appropriate quality of service guarantees for all virtual hosts. In such an embodiment, the present invention is in the form of a server application program. Unlike other embodiments of the present invention, in order to utilize this embodiment providers of virtual host services must upgrade their virtual host system.

[0019] In this embodiment, the virtual host server program stores in computer memory a table including appropriate quality of service guarantees. The server program then receives all client requests for communication with virtual hosts. When utilizing the fork after accept method, the server first receives a client request to communicate with a virtual host, and then creates a child process to manage communication between the virtual host and the client. Next, the server obtains the quality of service parameters associated with the virtual host, and guarantees, to the child process, the quality of service appropriate for the virtual host.

[0020] When utilizing the fork before accept method, the server first creates, a plurality of child processes to manage communication requests made to virtual hosts by clients. When the server receives a communication request made to a select one of the virtual hosts by a client, the server obtains the quality of service parameters associated with the virtual host, and guarantees the associated quality of service to the child process that will manage the communication.

[0021] As here summarized, the present invention makes appropriate quality of service guarantees to a plurality of virtual hosts comprising a single server application program. Of course, the present invention can also be utilized to make appropriate quality of service guarantees to a plurality of virtual processes of any nature comprising a single application program.

[0022] The problem described in the foregoing may be embodied on a computer readable medium such as

an optical or hard disc, or it may be embodied on or in an electromagnetic signal such as a TCP/IP packet, and optical pulse or any other form of carrier.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] FIG. 1 is a block diagram illustrating a high level overview of a system for setting quality of service guarantees for virtual hosts in accordance with one embodiment of the present invention.

[0024] FIG. 2 is a block diagram illustrating a system for setting quality of service guarantees for virtual hosts of a virtual hosting system that utilizes a fork before accept method.

[0025] FIG. 3 is a block diagram illustrating a system for setting quality of service guarantees for virtual hosts of a virtual hosting system that utilizes a fork after accept method.

[0026] FIG. 4A, FIG. 4B, FIG. 4C, and FIG. 4D are block diagrams illustrating, in greater detail, select features depicted in FIG. 3. FIG. 4A illustrates pointers inserted into the operating system. FIG. 4B illustrates copies of pointers to operating system calls saved to computer memory. FIG. 4C illustrates the content of object code inserted into the operating system. FIG. 4D illustrates the content of association tables.

[0027] FIG. 5 is a block diagram illustrating a system for setting quality of service guarantees for virtual hosts of a name-based virtual hosting system according to one embodiment of the present invention.

[0028] FIG. 6A, FIG. 6B, and FIG. 6C are block diagrams illustrating, in greater detail, select features depicted in FIG. 5. FIG. 6A illustrates pointers inserted into the operating system. FIG. 6B illustrates copies of pointers to operating system calls saved to computer memory. FIG. 6C illustrates the content of object code inserted into the operating system.

[0029] FIG. 7 is a block diagram illustrating an embodiment in which the present invention comprises a stand alone virtual host system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

SYSTEM OVERVIEW

[0030] FIG. 1 presents a high level overview of a system for setting quality of service guarantees for virtual hosts in accordance with a preferred embodiment of the present invention. A single, physical host computer 101 contains computer memory 103, conventional processor(s), networking interfaces, and input/output devices (not shown) . An operating system 105, a virtual host server application program 107, and a plurality of child processes 109 of the server application program 107 r - side in the computer memory 103. For purposes of example, FIG. 1 illustrates three child processes 109 of the virtual host server application program 107: a first child process 109A, a second child process 109B, and a third child process 109C. It is to be understood that more or fewer child processes 109 can reside in th computer memory 103 as desired.

[0031] Client computers 111 send requests 113 to a plurality of virtual hosts 115 which are supported by th server 107. All such requests are received by the server application program 107 and are processed by the child processes 109. For purposes of example, FIG. 1 illustrates three client computers 111 (client computer 111A, client computer 111B, and client computer 111C), each making separate requests 113 to a separate virtual host 115. It is to be understood that more or fewer client computers 111 can make more or fewer requests 113 to more or fewer virtual hosts 115. It is to be understood that the clients 111 are typically remote from the server 107 and physical host computer 101.

[0032] A virtual host quality of service application program 117 executes in the computer memory 103 of the physical host computer 101. The virtual host quality of service application program 117 inserts a quality of service parameter table 119 and object code 121 into the operating system 105 in the computer memory 103. Additionally, the virtual host quality of service application program 117 inserts, into the interrupt vector table 123 of the operating system 105, a pointer 125 to the object code 121. The object code 121 reads the quality of service parameter table 119, and calls the quality of service manager 127 of the operating system 105 to set appropriate quality of service guarantees for the virtual hosts 115 serviced by the virtual host server application program 107. The virtual hosts 115 then transmit responses 129 to the client computers 111. The features and functionality depicted in FIG. 1 are described in detail below.

PROVIDING QUALITY OF SERVICE GUARANTEES TO NETWORK ADDRESS-BASED VIRTUAL HOST SYSTEMS

I. Virtual Host Systems Utilizing the Fork Before Accept Method

[0033] FIG. 2 illustrates a virtual host application program 117 for providing quality of service guarantees to virtual hosts 115 of a server program 107 that utilizes the fork before accept virtual hosting method. As explained previously, clients 111 make requests 113 to specific virtual hosts 115. In a network address-based system, a virtual host 115 is identified by a network address 201. In other words, clients 111 transmit communication requests 113 to individual network addresses 201. Multiple network addresses 201 are assigned to the single, physical host computer 101, and communication requests thereto are serviced by the virtual host server program 107.

[0034] A virtual host server 107 that utilizes th fork before accept virtual hosting method generates a plurality of child processes 109, each of which waits for an

incoming communication request 113. When a child process 109 receives a request 113, the child process 109 establishes a communication channel between itself and the requesting client 111. The child process 109 proceeds to use the communication channel to service the request 113 that was made by the client 111 to the network address 201.

[0035] During the time that the child process 109 services the communication request 113 to the specific network address 201, it is desirable for the child process 109 to be guaranteed a specific quality of service according to parameters for the virtual host 115 with which the network address 201 is associated. For this reason, the present invention detects when a child process 109 is accepting a communication request 113, and sets quality of service guarantees for the child process 109 based on parameters for the network address 201 to which the request 113 is directed. When the child process 109 finishes a communication session initiated by a request 113 to a specific network address 201, the child process 109 closes the established communication channel and waits for another incoming request 113. The next request 109 the child process 109 receives may be directed to a different network address 201, and hence may require that different quality of service parameters be set for the child process 109. Therefore, the present invention detects when the child process 109 establishes a communication channel to service the new request 113, and proceeds to request appropriate quality of service guarantees for the target network address 201.

[0036] In the embodiment of the present invention depicted in FIG. 2, a virtual host quality of service application program 117 executes in the computer memory 103. The quality of service program 117 inserts a quality of service parameter table 119 into the operating system 105 of the host computer 101. The quality of service table 119 contains quality of service parameters for each network address 201 associated with one of the virtual hosts 115 serviced by the virtual host server 107. The quality of service program 117 utilizes techniques known in the art to insert the table 119 into the operating system 105. In a preferred embodiment, the present invention dynamically links a module to an operating system kernel, while the kernel is active. The module is preferably in the form of object code comprising an empty quality of service table 119, and subroutines to add, modify, and delete quality of service parameters for different virtual hosts. The quality of service application program 117 then calls the appropriate subroutine to add the quality of service parameters for the virtual hosts 115 serviced by the server program 107. The quality of service program 117 utilizes the subroutines to add, modify, and delete quality of service parameters as desired. In alternative embodiments, the module contains additional subroutines, or only a subset of the subroutines listed abov . In one alternative embodiment, the table 119 is first filled with quality of service parameters and then linked to kernel as a module. In an alternative embodiment, the quality of servic table 119 is stored outside of the operating system 105 in computer memory 103.

[0037] The quality of service program 117 also inserts object code 121 into the operating system 105. Preferably, the object code is dynamically linked to the operating system kernel as a module. The contents and execution of the object code 121 are discussed in detail below. In an alternative embodiment, the object code 121 resides outside of the operating system 105 in computer memory 103. The quality of service program 117 makes a copy 203 of an internal operating system pointer to the operating system function that is called to establish a communication channel. In an alternative embodiment, the copy 203 is made by a module linked to the operating system kernel. In the UNIX operating system, this function is the operating system accept function 205, as pictured in FIG. 2. The copy 203 of the pointer to the system accept function 205 is stored in conventional computer memory 103. In another embodiment, the copy 203 of the pointer is stored in the operating system 105.

[0038] It is to be understood that the name of the system function that is called to establish a communication channel can vary from operating system to operating system. The present invention is not limited to any specific operating system, or to any specific operating system function name. Furthermore, some operating systems include more than one function that establishes a communication channel. Embodiments of the present invention targeted for such operating systems create copies 203 of the pointers to all such operating system functions.

[0039] The pointer to the system accept function 205 is located in the operating system 105 interrupt vector table 123. It is to be understood that the term "interrupt vector table" as used herein denotes an area in operating system memory in which are stored the addresses of operating system functions (system calls). In the UNIX operating system, this part of the operating system is called the "interrupt vector table," and that term is used in this specification. Other operating systems employ different terminology to denote the same system component. An interrupt vector table by any other name is still within the scope of the present invention.

[0040] The quality of service program 117 replaces the pointer to the system accept function 205 with a pointer 125 to the inserted object code 121, such that when the system accept function 205 is called, the inserted object code 121 is executed instead. In another embodiment, this pointer replacement is executed by a module linked to the operating system 105 kernel. Executing alternative code when a system call is made comprises intercepting the system call. The steps of inserting object code 121 into the operating system 105, making a copy 203 of an operating system pointer, and replacing the operating system pointer with a pointer 125

to the inserted object code facilitate interception of a system call.

[0041] When a call is made to the system accept function 205, the operating system 105 uses the pointer 125 in the interrupt vector table 123 to execute th obj ct code 121. Th object code 121 first utilizes the saved copy 203 of the pointer to the system accept function 205 to call the system accept function 205. The system accept function 205 executes, thereby establishing the communication channel. When the accept function 205 terminates, the object code 121 continues to execute. The object code 121 determines if the communication channel that was established is to one of the network addresses 201 associated with one of the virtual hosts 115 serviced by the server program 107. Preferably, this determination is made by reading an operating system variable 207 that identifies the network address 201 associated with the communication channel. In UNIX, this system variable 207 is called "local_name," and is set by the accept function 205. In other operating systems the variable 207 is denoted by a different name. Alternative embodiments of the present invention utilize other methods to determine the network address 201 to which the communication channel was established, for example reading a return value of a system function, polling a system communication manager, or other similar mechanisms as desired.

[0042] Where the communication channel that was established is not to one of the network addresses 201 associated with one of the virtual hosts 115, the object code 121 simply exits. The channel will not be used for communication to a virtual host 115, and thus no quality of service guarantees need to be made by the present invention. Where the communication channel is to one of the network addresses 201 associated with one of the virtual hosts 115, the object code 121 requests, from the operating system 105, an appropriate quality of service guarantee for the process that called the system accept function 205.

[0043] To set the quality of service guarantees, the object code 121 reads the quality of service table 119, and locates the quality of service parameters for the virtual host 115 associated with the network address 201 to which the communication channel was established. The object code 121 then calls the operating system quality of service manager 127 to request an appropriate quality of service guarantee to the child process 109 that called the system accept function 205. The appropriate quality of service to guarantee is that specified by the quality of service parameters for the virtual host 115 to which the communication channel has been established. Once the quality of service guarantee has been requested, the object code 121 exits, and the child process 109 sends a response 129 to the client 111 via the established communication channel. Communication between the client 111 and the virtual host 115 proceeds over the communication channel. The communication is managed by the child process 109, which has been guaranteed appropriate quality of service.

II. Virtual Host Systems Utilizing the Fork After Accept Method

[0044] FIG. 3 illustrates a virtual host application program 117 for providing quality of service guarantees to virtual hosts 115 of a server program 107 that utilizes the fork after accept virtual hosting method. As explained previously, communication requests 113 to specific network addresses 201 are made by client computers 111 and are serviced by the virtual host server program 107. A virtual host server 107 that utilizes the fork after accept virtual hosting method accepts a communication request 113, establishes a communication channel between the client computer 111 and the network address 201, and then generates a child process 109. The child processes 109 uses the communication channel created by the server 107 to service the request 113 that was made by the client 111 to the network address 201. During the time the that the child process 109 services the communication request 113 to the specific network address 201, it is desirable for the child process 109 to be guaranteed quality of service according to parameters for the virtual host 115 with which the network address 201 is associated.

[0045] In the embodiment of the present invention depicted in FIG. 3, a virtual host quality of server program 117 executes in the computer memory 103. The quality of service program 117 inserts a quality of service parameter table 119 and object code 121 into the operating system 105 of the host computer 101, in the manner described above. The quality of service program 117 also inserts object code 121 into the operating system 105. As described above, preferably the object code is dynamically linked to the operating system kernel as a module. The contents and execution of the object code 121 are discussed in detail below.

[0046] The quality of service program 117 makes copies 203 of multiple internal operating system pointers to operating system functions. In an alternative embodiment, the copies 203 are made by a module linked to the operating system kernel. The present invention makes copies 203 of the pointers to the following operating system functions: system functions to establish a communication channel (for example, the UNIX accept function) 205, system functions to generate a child process (for example, the UNIX fork function) 301, system functions to copy a file or to copy a file descriptor (for example, the UNIX dup and dup2 functions), and system functions to read data from a file (for example, the UNIX read function). The copies of 203 of the pointers to the system functions are preferably stored in conventional computer memory 103. Alternatively, the copies 203 are stored in th operating system 105.

[0047] It is to b understood that the names of the relevant system functions can vary from operating system to operating system. The present invention is not limited

to any specific operating system, or to any specific operating system function names. Furthermore, some operating systems include more than one function which performs the above described functionality. Embodiments of the present invention targeted for such operating systems create copies 203 of the pointers to the appropriate operating system functions. For purposes of example, this specification refers to the system accept function 205, the system fork function 301, the system dup function 303, and the system read function 305, as depicted in FIG. 3. It is to be understood that references to the system accept function 205 apply to all system functions to establish a communication channel, references to the system fork function 301 apply to all system functions to generate a child process, references to the system dup function 303 apply to all system functions to copy a file or to copy a file descriptor, and references to the system read function 305 apply to all system functions to read data from a file or to read data from a communication channel.

[0048] The pointers to the above listed system functions are located in the operating system 105 interrupt vector table 123. The quality of service program 117 replaces the pointers with pointers 125 to inserted object code 121, such that when a targeted system function is called, inserted object code 121 is executed instead. In an alternative embodiment, the replacement is executed by a module linked to the operating system kernel. The embodiment of the present invention depicted in FIG. 3 intercepts the system accept function 205, the system fork function 301, the system dup function 303 and the system read function 305.

[0049] When a call is made to an intercepted function 205, inserted object code 121 executes. This inserted object code 121 is called as a wrapper. For example, the accept wrapper is the inserted object code 121 that executes when the system accept function 205 is called. FIG. 4C illustrates the content of the inserted object code 121 in the embodiment of FIG. 3. The object code 121 contains an accept wrapper 417, a fork wrapper 419, a dup wrapper 421, and a read wrapper 423. FIG. 4A illustrates pointers 125 inserted into the interrupt vector table: a pointer 401 to the accept wrapper 417, a pointer 403 to the fork wrapper 419, a pointer 405 to the dup wrapper 421, and a pointer 407 to the read wrapper 423. FIG. 4B illustrates the copies 203 of pointers: a copy of the pointer 409 to the system accept function 205, a copy of the pointer 411 to the system fork function 301, a copy of the pointer 413 to the system dup function 303, a copy of the pointer 415 to the system read function 305.

[0050] Returning to FIG. 3, whenever a process establishes a communication channel, the accept wrapper 417 executes. The accept wrapper 417 first utilizes the saved copy of the pointer 409 to the system accept function 205 to call the system accept function 205. The system accept function 205 executes, thereby establishing the communication channel. When the accept function 205 terminates, the accept wrapper 417 continues to execute. The accept wrapper 417 determines if the communication channel that was established is to one of th network addresses 201 associated with one of the virtual hosts 115 serviced by the server program 107. As described above, this determination is pr ferably made by reading an operating system variable 207 that identifies the network address 201 associated with the communication channel.

[0051] Where the communication channel that was established is not to one of the network addresses 201 associated with one of the virtual hosts 115, the accept wrapper 417 exits. The channel will not be used for communication to a virtual host 115, and thus no quality of service guarantees need to be made by the present invention. Where the communication channel is to one of the network addresses 201 associated with one of the virtual hosts 115, the present invention requests an appropriate quality of service guarantee for the child process 109 that will manage the communication. Because the child process 109 has not yet been generated by the server application 107, the present invention stores select information in a set of association tables 307. The information stored will allow the present invention to later identify the child process 109 managing the communication, and set appropriate quality of service for that child process 109. Preferably, the association tables 307 reside in conventional computer memory 103. In alternative embodiments, the association tables are inserted into the operating system 105. FIG. 4D illustrates the content of one embodiment of the association tables 307. Preferably, there are three association tables: an application program-communication channel association table 425, an application program-child process association table 427, and a file descriptor-file descriptor copy association table 429.

[0052] Returning to FIG. 3, the accept wrapper 417 isolates an identifier of the process that established the communication channel. Preferably, the identifier comprises an operating system 105 assigned process identification (PID) of the process that called the accept function 205. The accept wrapper 417 also isolates an identifier of the communication channel itself. In UNIX, as well as in many other operating systems, a communication channel (socket) is identified by a file descriptor. Embodiments of the present invention targeted for such operating systems isolate the file descriptor associated with the communication channel. Other operating systems employ alternative mechanisms to identify a communication channel, and corresponding embodiments of the present invention isolate appropriate identifiers accordingly.

[0053] The present invention stores, in the application program-communication channel association table 425, the process identifier, the file descriptor (or alternatively other identifier) associated with the communication channel, and the network address 201 to which th communication channel was established. Thus, the applica-

tion program-communication channel association table 425 contains, for each communication channel established to one of th network addresses 201 associated with one of the virtual hosts 115, a record that a specific process established a specific communication channel between a client 111 and a sp cific n twork address 201. Once these associations are stored, the accept wrapper 417 exits.

[0054] Recall that a virtual host server 107 that utilizes the fork after accept method will, after establishing a communication channel between a client 111 and a virtual host 115, generate a child process 109 to manage the communication. Whenever any application generates a child process 109, the fork wrapper 419 executes. The fork wrapper 419 first uses the copy of the pointer 403 to the system fork function 301 to call the system fork function 301. The system fork function 301 executes and generates a child process 109. When the system fork function 301 exits, the fork wrapper 419 continues to execute. The fork wrapper 419 stores, in the application program-child process association table 427, the process identification (or alternative identifier) for the application program that called the system fork function 301, as well as the process identification (or alternative identifier) for the generated child process 109. Thus, the application program-child process association table 427 contains associations between all application programs and all child processes 109 thereof.

[0055] Whenever any process copies a file, or a descriptor of a file, the dup wrapper 421 executes. The dup wrapper 421 utilizes the copy of the pointer 413 to the system dup function 303 to execute the system dup function 303. The system dup function 303 executes and creates a copy of a file (or a copy of a file descriptor). When the system dup function 303 exits, the dup wrapper 421 continues to execute. The dup wrapper 421 stores, in the file descriptor-file descriptor copy association table 429, the association between the file descriptor of the file copied by the system dup function 303 and the file descriptor of the copy thereof. Where only a file descriptor was copied by the system dup function 303, the dup wrapper 421 stores the association between the source file descriptor and the copy of the source file descriptor. Thus, for every copied file descriptor, the file descriptor-file descriptor copy association table 429 contains an association between the original file descriptor and the copy thereof.

[0056] Whenever a process calls the system read function 305, the read wrapper 423 executes. The read wrapper 423 checks the association tables 307 to determine whether the process that called the system read function 305 is a child process 109 of an application program that established a communication channel to a virtual host 115. To make this determination, the read wrapper 423 first checks the application program-child process association table 427 to determine if the process that called the system read function 305 is a child process 109 of another process. If the process that called the system read function 305 is a child process 109, the read wrapper 423 checks the application program-communication channel association table 425 to determine if the parent process establish d a communication channel to a n twork address 201 associated with one of the virtual hosts 115.

[0057] If the process that called the system read function 305 is a child process 109 of a parent process that established a communication channel to a virtual host 115, the read wrapper 423 determines if the child process 109 called the system read function 305 in order to read from the communication channel to the virtual host 115 established by the parent process. To make this determination, the read wrapper 423 compares the file descriptor that the system read function 305 was called to read from with the file descriptor, in the application program-communication channel association table 425, associated with the communication channel established by the parent process to the virtual host 115.

[0058] If the file descriptor of the read function 305 is not identical to the file descriptor associated with the communication channel, the read wrapper 423 checks the file descriptor-file descriptor copy association table 429 to determine whether the file descriptor is a copy of the file descriptor associated with the communication channel. Recall that the file descriptor-file descriptor copy association table 429 contains associations between all file descriptors and copies thereof. The read wrapper 423 examines the table to determine whether the file descriptor is a copy of another file descriptor, and if so whether the source file descriptor is the descriptor associated with the communication channel to the virtual host 115 established by the parent process.

[0059] If the file descriptor is the descriptor of the communication channel (or a copy thereof) the read wrapper 423 examines the application program-child process 425 to determine to which network address 201 the communication channel was established by the server program 107.

[0060] If the process that called the system read function 305 is not a child process 109 of an application program that established a communication channel to a virtual host 115, or if the process is not reading from the communication channel to a virtual host 115 established by the parent process, the read wrapper 423 uses the copy of the pointer 415 to the system read function 305 to call the system read function 305, which proceeds to execute. If, on the other hand, the process that called the system read function 305 is a child process 109 of an application program that established a communication channel to a virtual host 115, and the process is reading from the communication channel to a virtual host 115 established by the parent process, the read wrapper 423 proceeds to set appropriate quality of service guarantees for the child process 109.

[0061] To set the quality of service guarantees, the read wrapper 423 reads the quality of servic table 119, and locates the quality of service parameters for the vir-

tual host 115 associated with the network address 201 to which the communication channel was established. The read wrapper 423 then calls the quality of service manager 127 to request an appropriate quality of service guarantee to the child process 109 that called the system read function 305. Once the quality of service guarantee has been requested, the read wrapper 423 utilizes the copy of the pointer 415 to the system read function 305 to call the system read function 305. The system read function 305 proceeds to read data from the communication channel. Then, the child process 109, which has been guaranteed appropriate quality of service, continues to manage communication between the client 111 and the virtual host 115 over the communication channel.

[0062] It is to be understood that in various embodiments of the present invention, the quality of service application program 117 intercepts not only the system read function 305, but other system functions that transport data through a communication channel. For example, different embodiments intercept a system write function, a system transport function, and other such functions as desired. The present invention includes in its scope intercepting any system function which transports data through a communication channel, and proceeding to set quality of service parameters for a child process in the manner detailed above in the description of intercepting the system read function 305.

PROVIDING QUALITY OF SERVICE GUARANTEES TO NAME-BASED VIRTUAL HOST SYSTEMS

[0063] FIG. 5 illustrates a virtual host application program 117 for providing quality of service guarantees to virtual hosts 115 of a name-based virtual hosting system. As explained previously, in a name-based system a virtual host 115 is identified not by a network address 201, but instead by a domain name 500. Thus, clients 111 transmit communication requests 113 to individual domain names 500. Multiple such domain names 500 are assigned to the single, physical host computer 101, and communication requests thereto are serviced by the virtual host server program 107.

[0064] In the embodiment of the present invention depicted in FIG. 5, a virtual host quality of service application program 117 executes in the computer memory 103 of the single, physical host computer 101. As in the embodiments of FIG's 2 and 3, the quality of service program 117 inserts a quality of service parameter table 119 into the operating system 105 of the host computer 101. In the embodiment of FIG. 5, the quality of service table 119 contains quality of service parameters for each domain name 500 associated with one of the virtual hosts 115 serviced by the virtual host server 107. As in the embodiments of FIG's 2 and 3, the quality of service program 117 also inserts object code 121 into the operating system 105. The nature of this object code 121 is explained in detail later in this specification. As in the embodiments of FIG's 2 and 3, the quality of service table 119, the object code 121, or both are alternatively stored in conventional computer memory 103.

[0065] The quality of service program 117 makes copies 203 of multiple internal operating system pointers to operating system functions. The program 117 makes copies 203 of pointers to the following operating system functions: the system functions to read data from a file (for example, the UNIX read function), and the system functions to close a file (for example, the UNIX close function). The copies of 203 of the pointers to the system functions are preferably stored conventional computer memory 103. Alternatively, the copies 203 of the pointers are inserted into the operating system 105. As explained above, it is to be understood that alternative embodiments of the present invention create copies 203 of pointers to other operating system functions as desired.

[0066] The pointers to the above listed system functions are located in the operating system 105 interrupt vector table 123. The quality of service program 117 (or alternatively a module linked to the operating system kernel) replaces the pointers with pointers 125 to inserted object code 121, such that when a targeted system function is called, inserted object code 121 is executed instead. The embodiment of the present invention depicted in FIG. 5 intercepts the system read function 305, and the system close function.

[0067] When a call is made to an intercepted function 205, inserted object code 121 executes. As explained earlier in this specification, this inserted object code 121 is known as a wrapper. FIG. 6C illustrates the content of the inserted object code 121 in the embodiment of the present invention depicted in FIG. 5. The inserted object code contains a read wrapper 423 and a close wrapper 605. FIG. 6A illustrates the pointers 125 inserted into the interrupt vector table 123: a pointer 407 to the read wrapper 423 and a pointer 601 to the close wrapper 605. FIG. 6B illustrates the copies 203 of pointers: a copy of the pointer 415 to the system read function 305 and a copy of the pointer 603 to the system close function 507.

[0068] Returning to FIG. 5, whenever a process calls the system read function 305, the read wrapper 423 executes. The read wrapper 423 uses the copy of the pointer 415 to the system read function 305 to call the system read function 305. The system read function 305 executes and reads data from a file descriptor. When the system read function 305 terminates, the read wrapper 423 resumes execution. The read wrapper 423 proceeds to determine if appropriate quality of service has already been guaranteed for the process that called the system read function 305. To make this determination, the read wrapper 423 examines a plurality of read flags 501, which may be stored in conventional computer memory 103 or inserted into the operating system 105. Each read flag 501 contains an identifier of a process (preferably a PID), an identifier of a communication channel (preferably a file descriptor), and an indication (flag) of whether quality of servic has been guaranteed

to th identified process for servicing communication via the channel. The read wrapper 423 xamines the read flags 501 to determine whether appropriate quality of service has already be n set for the combination of the process that called the read function 423 and the file descriptor from which that process is attempting to read data. If quality of service has already been set, the read wrapper 423 simply exits. On the other hand, if quality of service has not been set, the read wrapper 423 determines if the process is servicing a virtual host 115.

[0069] To so determine, the read wrapper 423 parses a read buffer 503 returned by the system read function 305. The read buffer 503 contains the name 505 of the file (or file equivalent, such as communication channel) from which the system read function 305 read data. The read wrapper 423 isolates this name 505, and compares it to the domain names 500 of the virtual hosts 115. If the name 505 is not a domain name 500 of a virtual host 115, the read wrapper 423 exits. The system read function 305 was not called to read data across a communication channel between a client 111 and virtual host 115, so the read wrapper 423 need not set quality of service for the process that called the read function 305. However, if the name 505 from the read buffer 505 is the domain name 500 of a virtual host 115, the read wrapper 423 must set quality of service guarantees for the process that called the read function 305.

[0070] To set the quality of service guarantees, the read wrapper 423 reads the quality of service table 119, and locates the quality of service parameters for the virtual host 115 associated with the name 505 from the read buffer 503. The read wrapper 423 then calls the quality of service manager 127 to request an appropriate quality of service guarantee to the process 109 that called the system read function 305. The read wrapper 423 then exits. The process 109 which called the system read function 305, which has been guaranteed appropriate quality of service, continues to manage communication between the client 111 and the virtual host 115 over the communication channel. It is to be understood that, as explained above, various embodiments of the present invention intercept various system calls that perform identical or similar tasks.

[0071] Whenever a process calls the system close function 507, the close wrapper 605 executes. The close wrapper 605 uses the copy of the pointer 603 to the system close function 507 to call the system close function 507. The system close function 507 executes and closes the file. When the system close function 507 terminates, the close wrapper 605 resumes execution. The close wrapper 605 checks the plurality of read flags 501 for a flag concerning the combination of the process that called the close function 507 and the file descriptor of the file that the process is attempting to close. If there is no read flag 501 for the combination, the close wrapper 605 simply exits. The combination is not one pertaining to virtual host 115 communication, and the close wrapper 605 need not adjust the read flags 501. On the other hand, if a read flag 501 exists for the combination, the close wrapper 605 modifies it to indicate that quality of service has not been set for the combination. This is useful because the process that called the close function 507 may later utilize the same file descriptor to manage communication between a client 111 and a different virtual host 115, requiring a different quality of service guarantee. Thus, it is useful that the read flag 501 indicates quality of service is not set.

PROVIDING QUALITY OF SERVICE GUARANTEES TO VIRTUAL HOST SYSTEMS BY A STAND ALONE SERVER APPLICATION PROGRAM

[0072] FIG. 7 illustrates an embodiment in which the present invention comprises a stand alone virtual host system 700 that sets appropriate quality of service guarantees for the virtual hosts 115 which it services. In the embodiment of FIG. 7, the present invention is in the form of a server application program 107 executing in the computer memory 103 of a stand alone host computer 101. The server program 107 stores a quality of service table 119 in computer memory 103. The table 119 contains quality of service parameters for all of the virtual hosts 115 serviced by the system. All client 111 requests 113 for communication with virtual hosts 115 are received by the server program 107. When utilizing the fork after accept method, the server program 107 first receives a client 111 request 113 to communicate with a virtual host 115, and then creates a child process 109 to manage communication between the virtual host 115 and the client 109. Next, the server program 107 obtains the quality of service parameters associated with the virtual host 115 from the quality of service table 119. The server application program 107 then calls the operating system quality of service manager 127 to request an appropriate quality of service guarantee for the child process 109. The child process proceeds 109 to manage the communication between the client 111 and the virtual host 115.

[0073] When utilizing the fork before accept method, the server program 107 first creates a plurality of child processes 109 to manage communication requests 113 made to virtual 115 hosts by clients 111. When the server program 107 receives a communication request 113 made to a select one of the virtual hosts 115 by a client 111, the server program 107 obtains the quality of service parameters associated with the virtual host 115 from the quality of service table 119. The server program 107 then calls the operating system quality of service manager 127 to request an set appropriate quality of service guarantee for the child process 109. The child process proceeds 109 to manage the communication between the client 111 and the virtual host 115.

[0074] It is to be understood that the present invention is not limited to guaranteeing appropriate quality of service to a plurality of virtual hosts. The present invention can be utilized to make appropriate quality of service

guarantees to a plurality of virtual processes of any nature.

Claims

1. A method in a computer system for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of virtual processes, the method comprising:

   storing a plurality of quality of service parameters associated with the virtual processes;
   intercepting a system call to the operating system that associates a child process of the second application program with a virtual process;
   utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the quality of service parameters associated with the virtual process; and
   allowing association between the child process and the virtual process.

2. The method of claim 1 further comprising:

   storing object code comprising instructions to request a quality of service guarantee to a child process according to associated quality of service parameters; and
   wherein intercepting comprises replacing a pointer to the system call with a pointer to the object code, such that calling the system call causes the object code to execute.

3. A method in a computer system for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of virtual hosts, the method comprising:

   storing quality of service parameters associated with the virtual hosts;
   intercepting system calls which pertain to communication between one of the virtual hosts and a client;
   obtaining the quality of service parameters associated with the virtual host;
   utilizing operating system resources to request a guarantee, to the child process of the second application program, of a quality of service according to the obtained quality of service parameters associated with the virtual host; and
   allowing communication b tween the virtual host and the client.

4. The method of claim 3 further comprising:

   storing object code comprising instructions to request a quality of service guarantee to a child process managing communication between a virtual host and a client, according to quality of service parameters associated with the virtual host; and
   wherein intercepting comprises replacing a pointer to the system call with a pointer to the inserted object code, such that calling the system call causes the object code to execute.

5. A method in a computer system for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of network addresses, the method comprising:

   storing quality of service parameters associated with the network addresses;
   intercepting system calls, made by the child processes, that establish a communication channel between a client and one of the network addresses;
   obtaining the quality of service parameters associated with the network address to which the communication channel is established;
   utilizing operating system resources to request a guarantee, to the child process that executed the intercepted system call, of a quality of service according to the obtained quality of service parameters associated with the network address; and
   allowing communication between the network address and the client, the communication being managed by the child process.

6. The method of claim 5 further comprising:

   storing object code;
   saving a copy of the pointer to the system call to intercept; and
   wherein intercepting comprises replacing a pointer to the system call with a pointer to the object code, such that calling the system call causes the object code to execute.

7. The method of claim 6 wherein execution of the object code comprises:

   using the saved copy of the pointer to transfer execution to the intercepted system call to establish the communication channel;
   determining whether the established communication channel is to one of the network addresses serviced by the second application program

and its child processes; and
in response to a determination that the communication channel is to one of the network addresses:

obtaining quality of service parameters for the network address; and
utilizing operating system resources to request a guarantee, to the child process that called the intercepted system call, of a quality of service according to the obtained quality of service parameters.

8. A method in a computer system for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of network addresses, the method comprising:

storing quality of service parameters associated with the network addresses;
intercepting system calls which establish a communication channel between a client and a network address;
allowing establishment of the communication channel;
storing an indicator of an association between an application program that called the intercepted system call, the established communication channel, and the network address to which the channel was established;
intercepting system calls which create a child process;
allowing creation of the child process;
storing an indicator of an association between an application program that called the intercepted system call and the created child process;
intercepting system calls which create a copy of a file descriptor;
allowing creation of the copy of the file descriptor;
storing an indicator of an association between the file descriptor and the created copy of the file descriptor;
intercepting system calls which transport data through a communication channel;
examining the stored association indicators to determine whether an application that called the intercepted system call is a child process of a parent application that established a communication channel to one of the network addresses;
in response to determining that th application that called the intercepted system call is a child process of a parent application that established a communication channel to one of the network addresses, examining the stored association indicators to determine whether the communication channel through which the intercepted system call transports data is the communication channel stablished by the parent application to one of th network address s;
in response to determining that the communication channel through which the intercepted system call transports data is the communication channel established by the parent application to one of the network addresses:

obtaining from the quality of service parameters associated with a network address to which the communication channel connects a client; and
utilizing operating system resources to request a guarantee to the child process that called the intercepted system call, of a quality of service according to the obtained quality of service parameters associated with the network address; and

allowing data to be transported through the communication channel.

9. A method in a computer system for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made on communication channels to a plurality of virtual hosts, the method comprising:

storing quality of service parameters associated with the virtual hosts;
storing indicators as to whether quality of service has been guaranteed to a child processes servicing a request to a virtual host on a communication channel;
intercepting system calls made by a child process to transport data through a communication channel between a virtual host and a client;
examining the stored indicators to determine if quality of service has been guaranteed to the child process;
in response to determining that quality of service has not been guaranteed to the child process:

parsing a buffer associated with the communication channel to isolate the name of the virtual host;
obtaining the quality of service parameters associated with the virtual host;
utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the ob-

tained quality of service parameters associat d with the virtual host; and

setting an indicator that quality of service has been guaranteed to th child process; and allowing communication between the virtual host and client, the communication being managed by the child process.

10. The method of claims 3 or 9 wherein storing the quality of service parameters comprises modifying an operating system to include a quality of service parameter table comprising quality of service parameters associated with the virtual hosts.

11. The method of claims 5 or 8 wherein storing the quality of service parameters comprises modifying an operating system to include a quality of service parameter table comprising quality of service parameters associated with the network addresses.

12. The method of claims 10 or 11 wherein modifying the operating system comprises linking a module to an operating system kernel, the module comprising the quality of service parameter table.

13. The method of claims 8 or 9 further comprising:

storing object code;
saving a copy of the pointer to the system call to intercept; and
wherein intercepting comprises replacing the pointer to the system call with a pointer to the inserted object code, such that calling the system call causes the object code to execute.

14. The method of claims 4,6 or 13 wherein storing object code comprises inserting the object code into the operating system.

15. The method of claim 14 wherein inserting object code into the operating system comprises linking a module to an operating system kernel, the module comprising the object code.

16. The method of claim 13, where claim 13 depends on claim 8, wherein the intercepting comprises intercepting system calls which establish a communication channel between a client and a network address, and execution of the object code further comprises:

allowing establishment of the communication channel by using the saved copy of the pointer to transfer execution to the intercepted system call;
storing an association between the application program and the established communication channel by storing in an association table an entry indicating an association between a process identification number of the application program that called the system call, a file descriptor describing the established communication channel, and the network address to which the channel was established.

17. The method of claim 13, where claim 13 depends on claim 8, wherein the intercepting comprises intercepting system calls which create a child process, and execution of the object code further comprises:

allowing creation of the child process by using the saved copy of the pointer to transfer execution to the intercepted system call; and
storing an association between the application program that called the system call and the created child process by storing in an association table an entry indicating an association between a process identification number of the application program that called the system call and a process identification number of the created child process.

18. The method of claim 9 wherein storing the indicators comprises modifying an operating system to include a quality an indicator table comprising the indicators as to whether quality of service has been guaranteed to a child processes servicing a request to a virtual host on a communication channel.

19. The method of claim 18 wherein modifying the operating system comprises linking a module to an operating system kernel, the module comprising the indicator table.

20. A method in a computer system for providing quality of service guarantees to child processes of an application program, wherein the application program and its child processes service communication requests made to a plurality of virtual hosts, the method comprising:

storing quality of service parameters associated with the virtual hosts;
receiving, by the application program, a communication request made to one of the virtual hosts by a client;
creating, by the application program, a child process to manage communication between the network address and the client;
obtaining from the quality of service table the quality of service parameters associated with the network address;
utilizing operating system resources to request a guarantee, to the child process, of a quality

of service according to the obtained quality of service parameters associated with the network address; and
allowing communication between the network address and the client, the communication being managed by the child process.

21. A method in a computer system for providing quality of service guarantees to child processes of an application program, wherein the application program and its child processes service communication requests made to a plurality of virtual hosts, the method comprising:

storing quality of service parameters associated with the virtual hosts;
creating, by the application program, a plurality of child processes to manage communication requests made to the virtual hosts by clients;
receiving, by the application program, a communication request made to one of the virtual hosts by a client;
obtaining from the quality of service table the quality of service parameters associated with the network address;
utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the obtained quality of service parameters associated with the network address; and
allowing communication between the network address and the client, the communication being managed by the child process.

22. A computer program product for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of virtual processes, the computer program product comprising:

program code for storing a plurality of quality of service parameters associated with the virtual processes;
program code for intercepting a system call to the operating system that associates a child process of the second application program with a virtual process;
program code for utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the quality of service parameters associated with the virtual process; and
program code for allowing association between the child process and the virtual process.

23. The computer program product of claim 22 further comprising:

program code for storing object code comprising instructions to request a quality of service guarantee to a child process according to associated quality of service parameters; and
wherein the program code for intercepting further comprises program code for replacing a pointer to the system call with a pointer to the object code, such that calling the system call causes the object code to execute.

24. A computer program product for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of virtual hosts, the computer program product comprising:

program code for storing quality of service parameters associated with the virtual hosts;
program code for intercepting system calls which pertain to communication between one of the virtual hosts and a client;
program code for obtaining the quality of service parameters associated with the virtual host;
program code for utilizing operating system resources to request a guarantee, to the child process of the second application program, of a quality of service according to the obtained quality of service parameters associated with the virtual host; and
program code for allowing communication between the virtual host and the client.

25. The computer program product of claim 24 further comprising:

program code for storing object code comprising instructions to request a quality of service guarantee to a child process managing communication between a virtual host and a client, according to quality of service parameters associated with the virtual host; and
wherein the program code for intercepting comprises program code for replacing a pointer to the system call with a pointer to the inserted object code, such that calling the system call causes the object code to execute.

26. A computer program product for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made to a plurality of network addresses, the computer program product comprising:

program code for storing quality of service parameters associated with the network addresses;
program code for intercepting system calls, made by the child processes, that establish a communication channel between a client and one of the network addresses;
program code for obtaining the quality of service parameters associated with the network address to which the communication channel is established;
program code for utilizing operating system resources to request a guarantee, to the child process that executed the intercepted system call, of a quality of service according to the obtained quality of service parameters associated with the network address; and
program code for allowing communication between the network address and the client, the communication being managed by the child process.

27. The computer program product of claim 26 further comprising:

program code for storing object code;
program code for saving a copy of the pointer to the system call to intercept; and
wherein the program code for intercepting comprises program code for replacing a pointer to the system call with a pointer to the object code, such that calling the system call causes the object code to execute.

28. The computer program product of claim 27 wherein the object code comprises:

program code for using the saved copy of the pointer to transfer execution to the intercepted system call to establish the communication channel;
program code for determining whether the established communication channel is to one of the network addresses serviced by the second application program and its child processes; and
program code for, in response to a determination that the communication channel is to one of the network addresses:

obtaining quality of service parameters for the network address; and
utilizing operating system resources to request a guarantee, to the child process that called the intercepted system call, of a quality of service according to the obtained quality of service parameters.

29. A computer program product for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child proc sses service requests made to a plurality of network addresses, the computer program product comprising:

program code for storing quality of service parameters associated with the network addresses;
program code for intercepting system calls which establish a communication channel between a client and a network address;
program code for allowing establishment of the communication channel;
program code for storing an indicator of an association between an application program that called the intercepted system call, the established communication channel, and the network address to which the channel was established;
program code for intercepting system calls which create a child process;
program code for allowing creation of the child process;
program code for storing an indicator of an association between an application program that called the intercepted system call and the created child process;
program code for intercepting system calls which create a copy of a file descriptor;
program code for allowing creation of the copy of the file descriptor;
program code for storing an indicator of an association between the file descriptor and the created copy of the file descriptor;
program code for intercepting system calls which transport data through a communication channel;
program code for examining the stored association indicators to determine whether an application that called the intercepted system call is a child process of a parent application that established a communication channel to one of the network addresses;
program code for, in response to determining that the application that called the intercepted system call is a child process of a parent application that established a communication channel to one of the network addresses, examining the stored association indicators to determine whether the communication channel through which the intercepted system call transports data is the communication channel established by the parent application to one of the network addresses;
program code for, in response to determining

that the communication channel through which the intercepted system call transports data is the communication channel established by the parent application to one of the network addresses:

obtaining from the quality of service parameters associated with a network address to which the communication channel connects a client; and
utilizing operating system resources to request a guarantee to the child process that called the intercepted system call, of a quality of service according to the obtained quality of service parameters associated with the network address; and

program code for allowing data to be transported through the communication channel.

30. A computer program product for providing, by a first application program, quality of service guarantees to child processes of a second application program, wherein the second application program and its child processes service requests made on communication channels to a plurality of virtual hosts, the computer program product comprising:

program code for storing quality of service parameters associated with the virtual hosts;
program code for storing indicators as to whether quality of service has been guaranteed to a child processes servicing a request to a virtual host on a communication channel;
program code for intercepting system calls made by a child process to transport data through a communication channel between a virtual host and a client;
program code for examining the stored indicators to determine if quality of service has been guaranteed to the child process;
program code for, in response to determining that quality of service has not been guaranteed to the child process:

parsing a buffer associated with the communication channel to isolate the name of the virtual host;
obtaining the quality of service parameters associated with the virtual host;
utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the obtained quality of service parameters associated with the virtual host; and
setting an indicator that quality of service has been guaranteed to the child process; and

program code for allowing communication between the virtual host and client, the communication being managed by the child process.

31. The computer program product of claim 30 wherein the program code for storing the indicators comprises program code for modifying an operating system to include a quality an indicator table comprising the indicators as to whether quality of service has been guaranteed to a child processes servicing a request to a virtual host on a communication channel.

32. The computer program product of claim 31 wherein the program code for modifying the operating system comprises program code for linking a module to an operating system kernel, the module comprising the indicator table.

33. The computer program product of claims 24 or 30 wherein the program code for storing the quality of service parameters comprises program code for modifying an operating system to include a quality of service parameter table comprising quality of service parameters associated with the virtual hosts.

34. The computer program product of claims 26 or 29 wherein the program code for storing the quality of service parameters comprises program code for modifying an operating system to include a quality of service parameter table comprising quality of service parameters associated with the network addresses.

35. The computer program product of claims 33 or 34 wherein the program code for modifying the operating system comprises program code for linking a module to an operating system kernel, the module comprising the quality of service parameter table.

36. The computer program product of claims 29 or 30 further comprising:

program code for storing object code;
program code for saving a copy of the pointer to the system call to intercept; and
wherein the program code for intercepting comprises program code for replacing the pointer to the system call with a pointer to the inserted object code, such that calling the system call causes the object code to execute.

37. The computer program product of claims 25, 27 or 36 wherein the program code for storing object code comprises program cod for inserting the object code into the operating system.

38. The computer program product of claim 37 wherein

the program code for inserting object code into the operating system comprises program code for linking a module to an operating system kernel, the module comprising the object code.

39. The computer program product of claim 36, where claim 36 depends on claim 29, wherein the program code for the intercepting comprises program code for intercepting system calls which establish a communication channel between a client and a network address, and the object code further comprises:

program code for allowing establishment of the communication channel by using the saved copy of the pointer to transfer execution to the intercepted system call; and
program code for storing an association between the application program and the established communication channel by storing in an association table an entry indicating an association between a process identification number of the application program that called the system call, a file descriptor describing the established communication channel, and the network address to which the channel was established.

40. The computer program product of claim 36, where claim 36 depends on claim 29, wherein the program code for the intercepting comprises program code for intercepting system calls which create a child process, and wherein the object code further comprises:

program code for allowing creation of the child process by using the saved copy of the pointer to transfer execution to the intercepted system call; and
program code for storing an association between the application program that called the system call and the created child process by storing in an association table an entry indicating an association between a process identification number of the application program that called the system call and a process identification number of the created child process.

41. A computer program product for providing quality of service guarantees to child processes of an application program, wherein the application program and its child processes service communication requests made to a plurality of virtual hosts, the computer program product comprising:

program code for storing quality of service parameters associated with the virtual hosts;
program code for receiving, by the application program, a communication request made to one of the virtual hosts by a client;
program code for creating, by the application program, a child process to manage communication between the network addr ss and the client;
program code for obtaining from the quality of service table the quality of service parameters associated with the network address;
program code for utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the obtained quality of service parameters associated with the network address; and
program code for allowing communication between the network address and the client, the communication being managed by the child process.

42. A computer program product for providing quality of service guarantees to child processes of an application program, wherein the application program and its child processes service communication requests made to a plurality of virtual hosts, the computer program product comprising:

program code for storing quality of service parameters associated with the virtual hosts;
program code for creating, by the application program, a plurality of child processes to manage communication requests made to the virtual hosts by clients;
program code for receiving, by the application program, a communication request made to one of the virtual hosts by a client;
program code for obtaining from the quality of service table the quality of service parameters associated with the network address;
program code for utilizing operating system resources to request a guarantee, to the child process, of a quality of service according to the obtained quality of service parameters associated with the network address; and
program code for allowing communication between the network address and the client, the communication being managed by the child process; and a computer readable medium on which the program codes are stored.

43. The computer program product of any one of claims 22 to 42, embodied upon a computer readable medium.

44. The computer program product of any one of claims 22 to 42, embodied upon, or within, an electromagnetic signal.

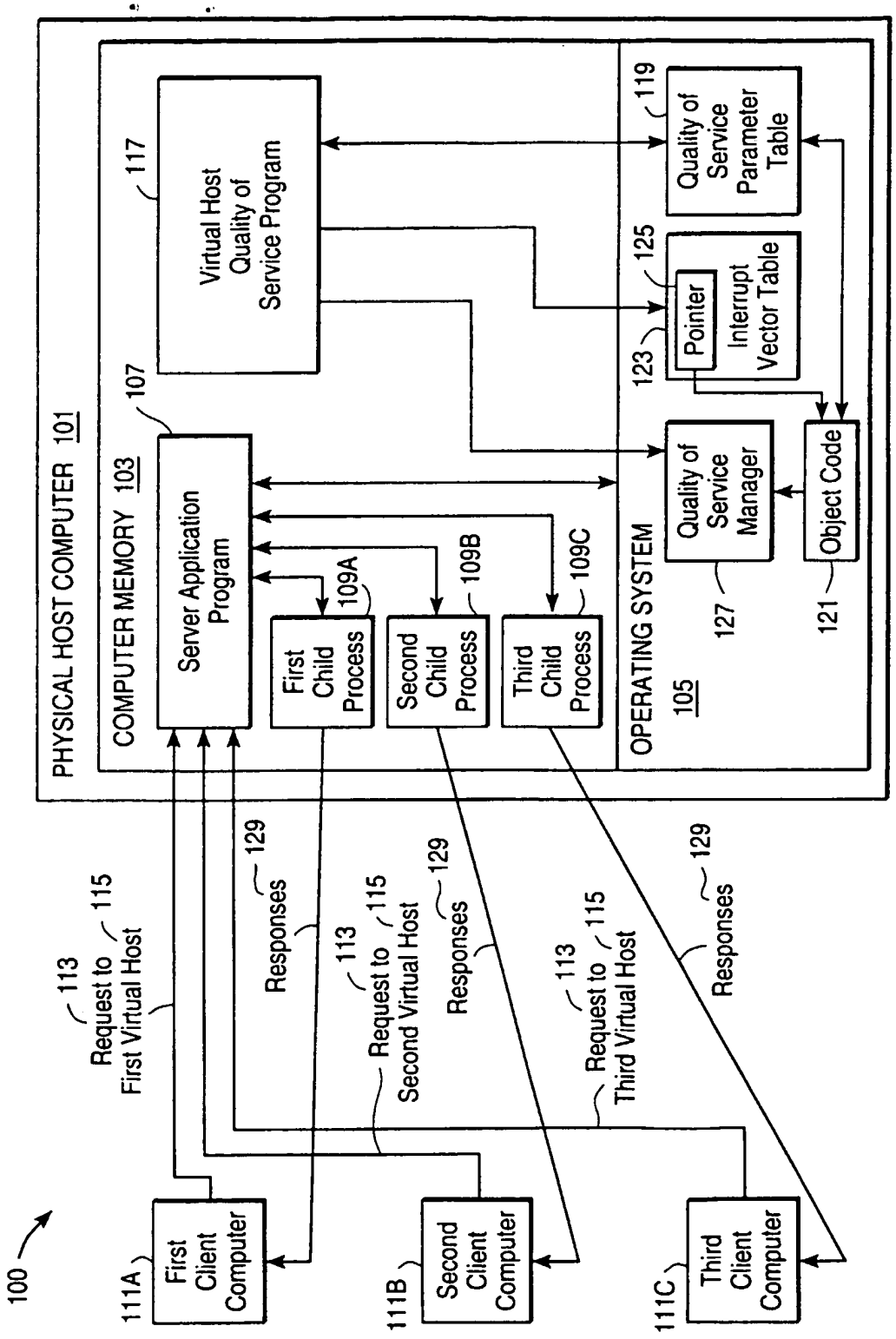
100
PHYSICAL HOST COMPUTER 101
COMPUTER MEMORY 103
Virtual Host Quality of Service Program
117
Server Application Program
107
First Child Process
109A
Second Child Process
109B
Third Child Process
109C
OPERATING SYSTEM
105
Quality of Service Parameter Table
119
Pointer
125
Interrupt Vector Table
123
Quality of Service Manager
127
Object Code
121
113
Request to First Virtual Host
115
Responses
129
Request to Second Virtual Host
Responses
Request to Third Virtual Host
Responses
First Client Computer
111A
Second Client Computer
111B
Third Client Computer
111C

FIG. 1

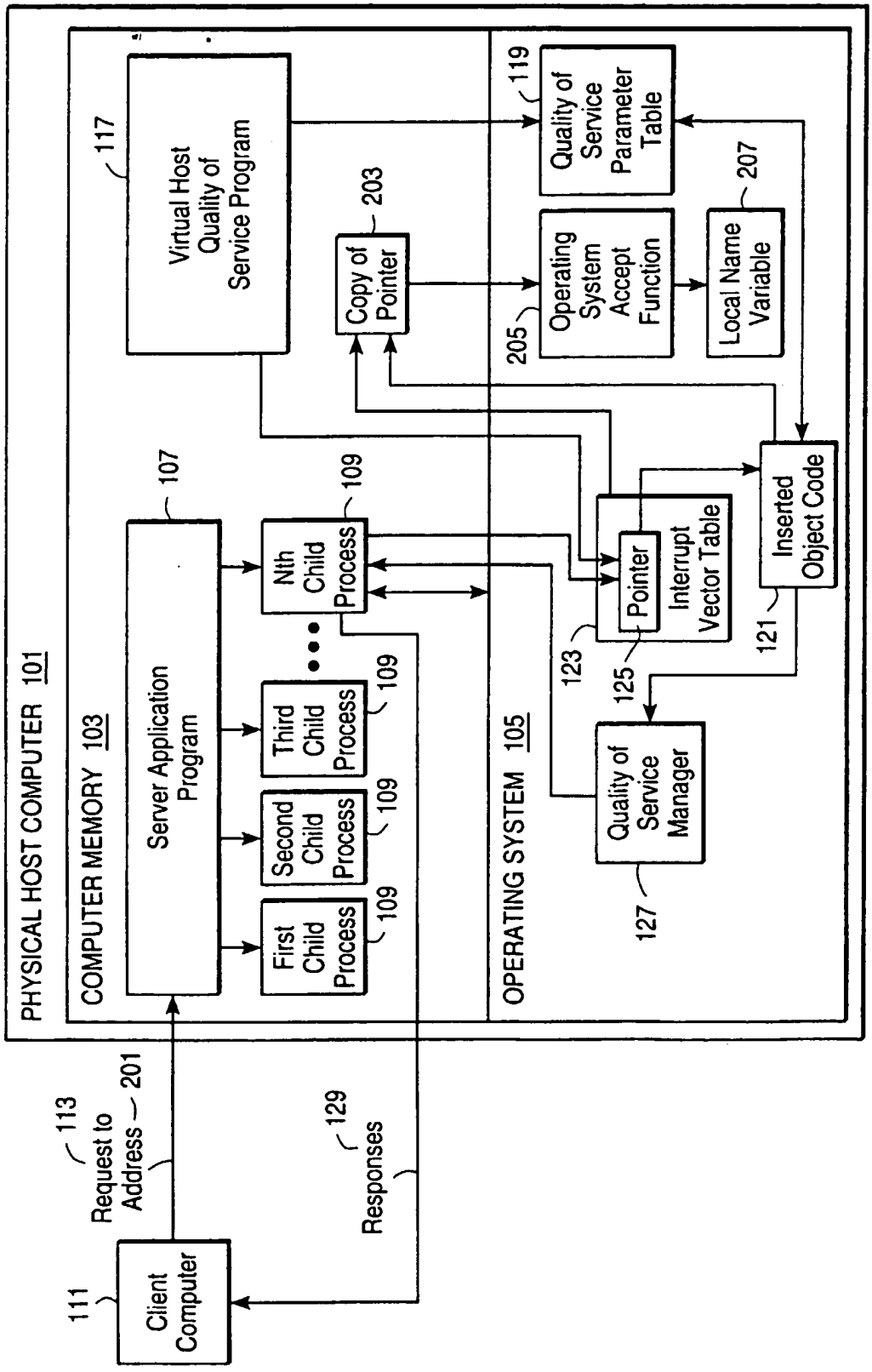
PHYSICAL HOST COMPUTER 101
COMPUTER MEMORY 103
OPERATING SYSTEM 105
Virtual Host Quality of Service Program
117
Quality of Service Parameter Table
119
Copy of Pointer
203
Operating System Accept Function
205
Local Name Variable
207
Server Application Program
107
Nth Child Process
109
First Child Process
Second Child Process
Third Child Process
Pointer
Interrupt Vector Table
123
Inserted Object Code
121
125
Quality of Service Manager
127
Client Computer
111
Request to Address
113
201
Responses
129

FIG. 2

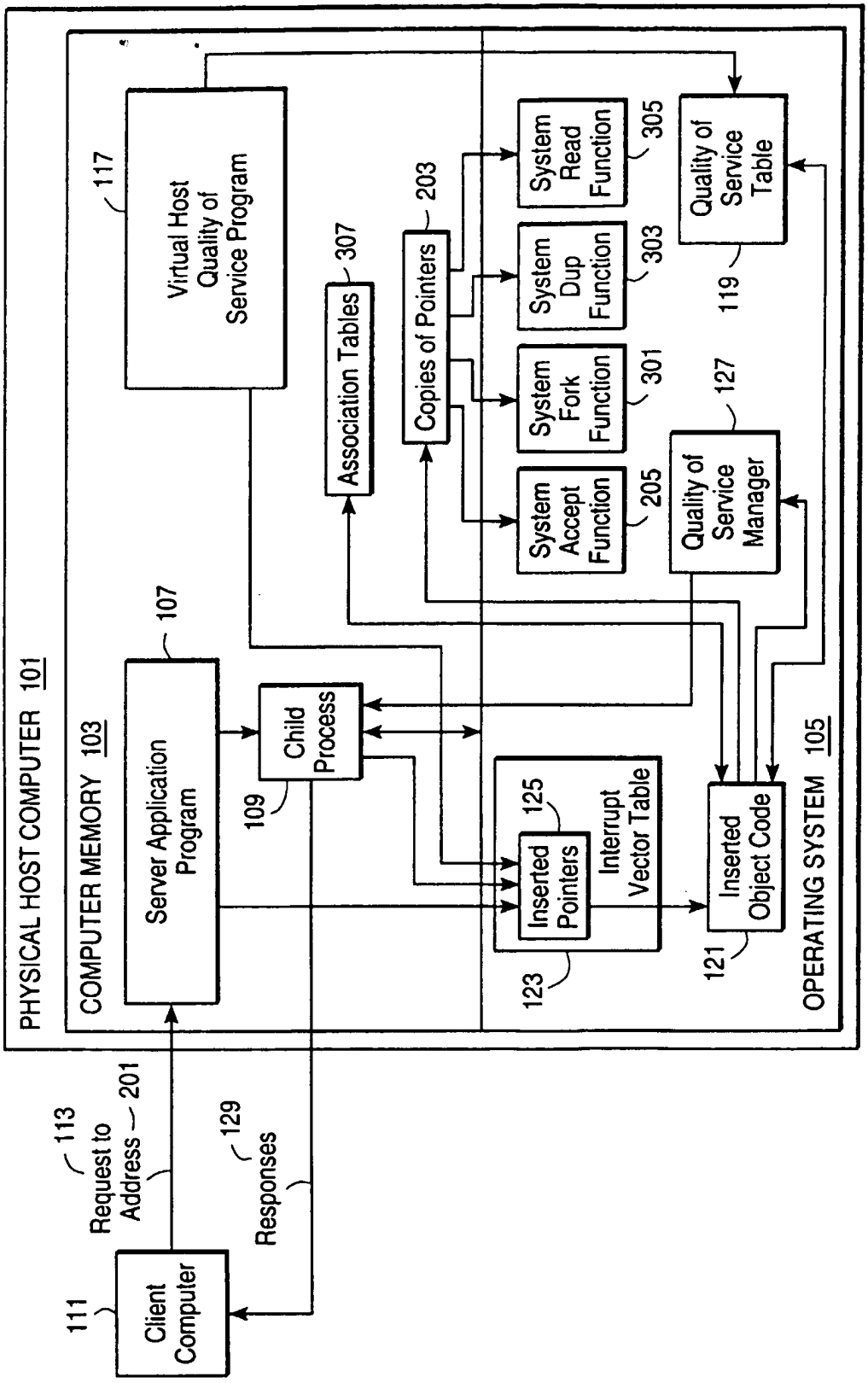
PHYSICAL HOST COMPUTER 101
COMPUTER MEMORY 103
OPERATING SYSTEM 105
Virtual Host Quality of Service Program
117
Server Application Program
107
Child Process
109
Association Tables
307
Copies of Pointers
203
System Accept Function
205
System Fork Function
301
System Dup Function
303
System Read Function
305
Quality of Service Manager
127
Quality of Service Table
119
Inserted Pointers
125
Interrupt Vector Table
123
Inserted Object Code
121
Client Computer
111
Request to Address
113
201
Responses
129

FIG. 3

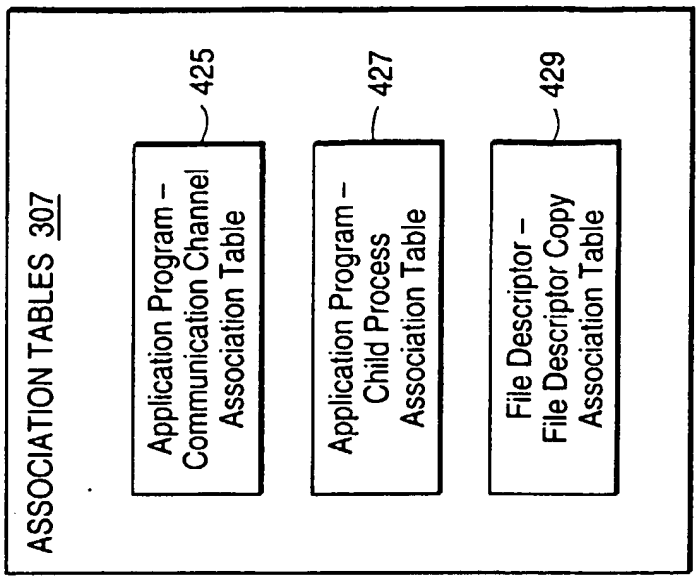

FIG. 4D

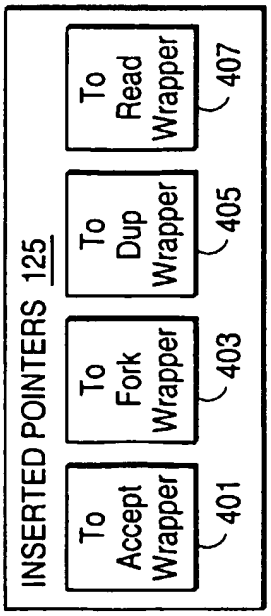

FIG. 4A

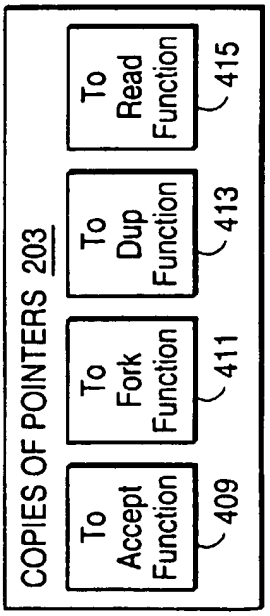

FIG. 4B

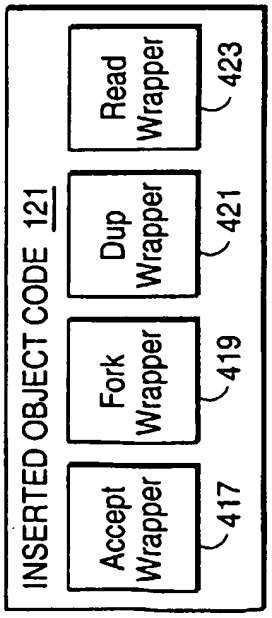

FIG. 4C

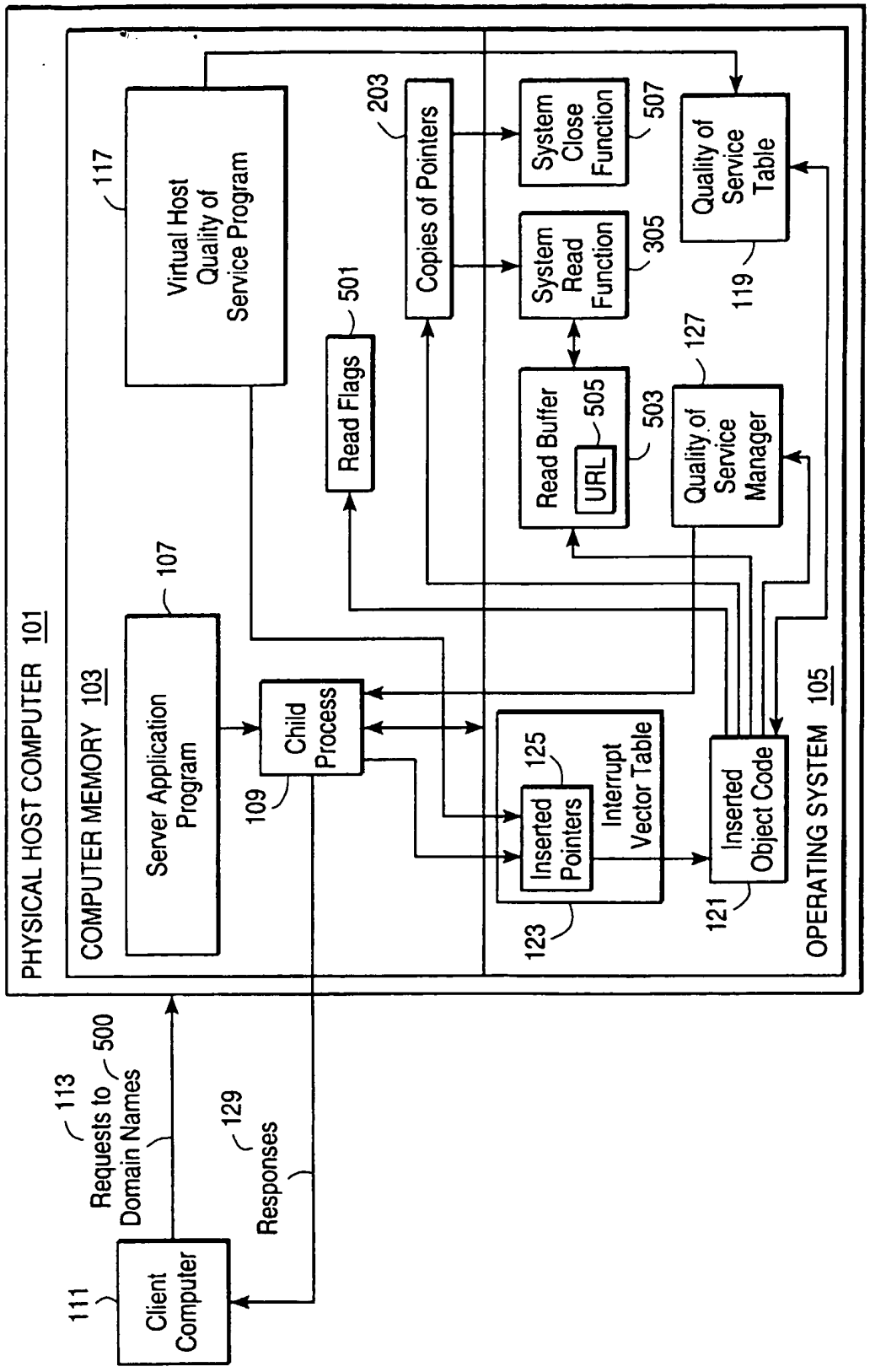
PHYSICAL HOST COMPUTER 101
COMPUTER MEMORY 103
OPERATING SYSTEM 105
Virtual Host Quality of Service Program
117
Server Application Program
107
Child Process
109
Read Flags
501
Copies of Pointers
203
System Close Function
507
System Read Function
305
Quality of Service Table
119
Read Buffer
URL
505
503
Quality of Service Manager
127
Inserted Pointers
125
Interrupt Vector Table
123
Inserted Object Code
121
Requests to Domain Names
113
500
Responses
129
Client Computer
111

FIG. 5

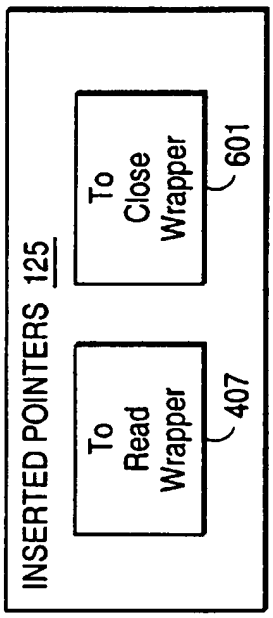

FIG. 6A

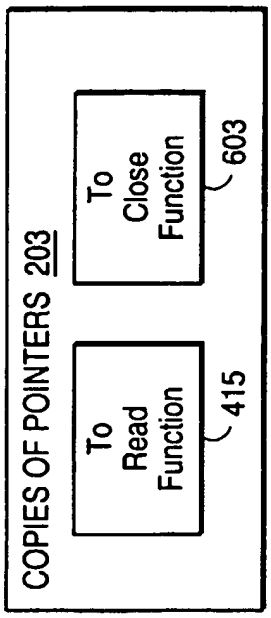

FIG. 6B

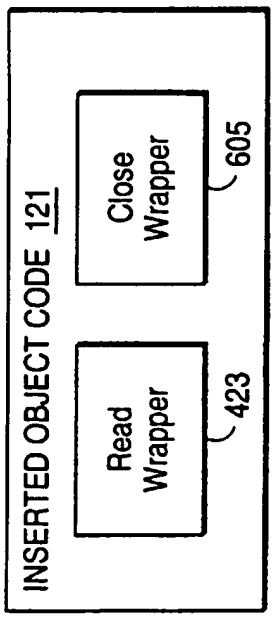

FIG. 6C

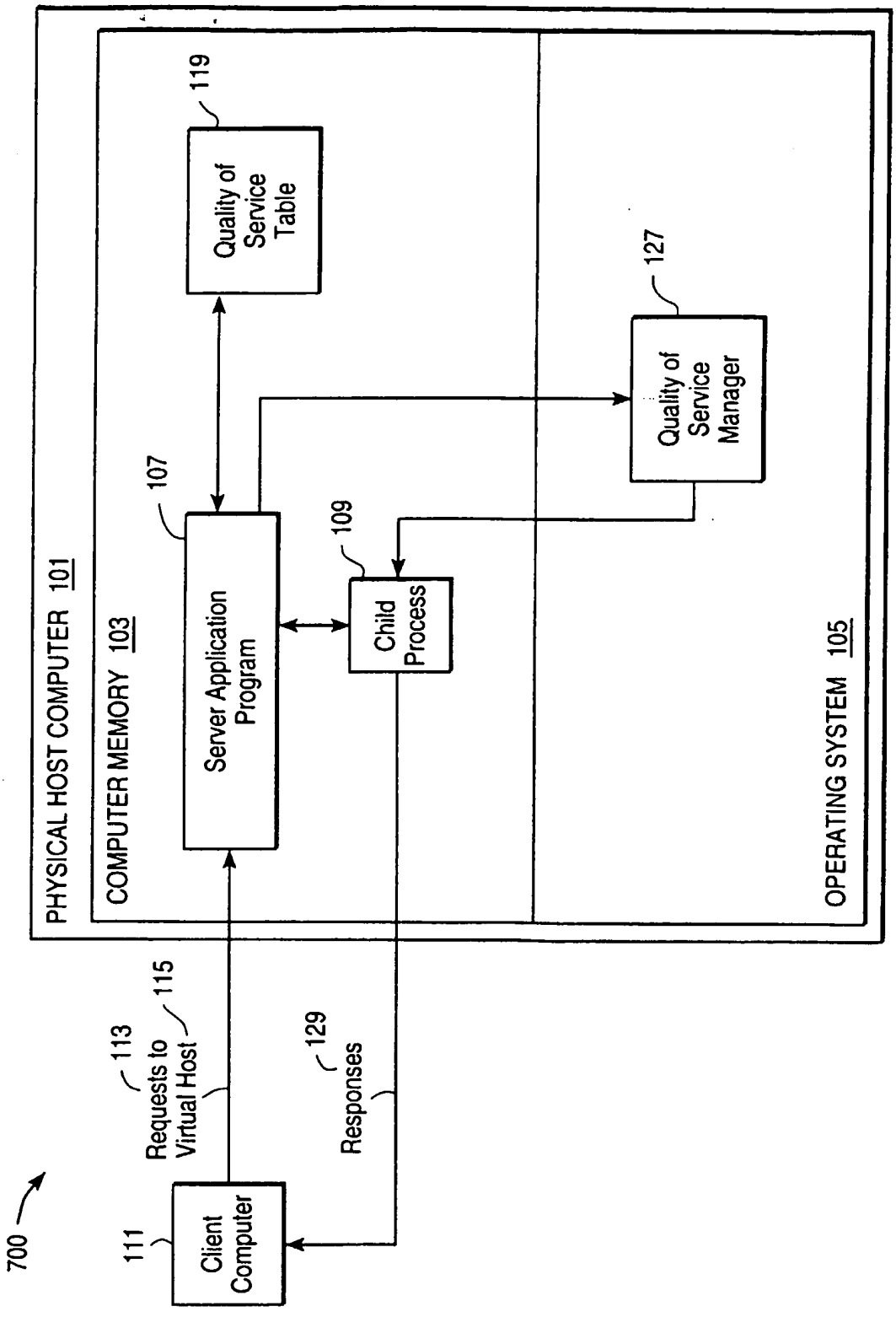
700
PHYSICAL HOST COMPUTER 101
COMPUTER MEMORY 103
OPERATING SYSTEM 105
Quality of Service Table
119
Quality of Service Manager
127
Server Application Program
107
Child Process
109
Requests to Virtual Host
113
115
Responses
129
Client Computer
111

FIG. 7